```
0000:            1 * FMON -- THE VCS SIDE OF THE DEBUGGER SYSTEM
0000:            2 * COPYRIGHT 1982 BY FROBCO
0000:            3 * ALL RIGHTS RESERVED
0000:            4 *
0000:            5 * VERSION 1.3 -- LAST MODIFIED 1/26/83
0000:            6 *
0000:            7 *
0000:            8 * SYSTEM DEFINITIONS
0000:            9 *
FFF1:           10 PSTAT   EQU  $FFF1     ; STATUS PORT
FFF2:           11 RDATA   EQU  $FFF2     ; DATA PORT FROM APPLE
FFF0:           12 WDATA   EQU  $FFF0     ; DATA PORT TO APPLE
0000:           13 *
FFF4:           14 F1REST  EQU  $FFF4     ; FIRST PART OF FLAG RESTORE
FFF5:           15 F2REST  EQU  $FFF5     ; LAST PART OF FLAG RESTORE
FFF6:           16 AREST   EQU  $FFF6     ; RESTORE A FROM HERE
FFF7:           17 XREST   EQU  $FFF7     ; RESTORE X FROM HERE
FFF8:           18 YREST   EQU  $FFF8     ; RESTORE Y FORM HERE
FFF9:           19 SREST   EQU  $FFF9     ; RESTORE S FORM HERE
0000:           20 *
0000:           21 *
0000:           22 *
0000:           23 * RAM DEFINITIONS
0000:           24 *
00E0:           25 RSAVE   EQU  $E0       ; PLACE TO START SAVEING RAM LOCATIONS
00F8:           26 RBASE   EQU  $F8       ; GENERAL 16 BIT POINTER
0000:           27 *
0000:           28 *
----- NEXT OBJECT FILE NAME IS FMON.OBJ0
FF00:           29         ORG  $FF00     ; FMON ADDRESS SPACE
FF00:           30 *
FF00:           31 *
FF00:           32 * THIS IS THE ENTRY POINT UPON BREAK POINT
FF00:           33 * THE SYSTEM SAVE THE STATE BY PASSING INFORMATION
FF00:           34 * OVER TO THE APPLE.
FF00:           35 *
FF00:           36 * THE TRICK HERE IS TO SAVE THE STATE OF THE N AND Z
FF00:           37 * FLAGS WITHOUT USING THE STACK WHICH MAY OR MAY NOT
FF00:           38 * BE SET UP AS A STACK AT THE TIME OF THE BREAK POINT
FF00:           39 *
FF00:           40 *
FF00:8D F0 FF   41 BREAK   STA  WDATA     ; SEND THE A REGISTER TO THE APPLE
FF03:D0 09      42         BNE  NOTZ      ; BRANCH IF Z FLAG NOT SET
FF05:AD F1 FF   43 BRKW1   LDA  PSTAT     ; ELSE SEND A CODE THAT WILL
FF08:10 FB      44         BPL  BRKW1     ; RESTORE N=0 AND Z=1 WHEN WE GET BACK
FF0A:A9 FF      45         LDA  #$FF      ; AND DO AN INC MEM. (FF WILL INC TO 0)
FF0C:D0 12      46         BNE  BRK1      ; BRANCH ALWAYS
FF0E:           47 *
FF0E:30 09      48 NOTZ    BMI  FNEG      ; KEEP GOING IF N=1
FF10:AD F1 FF   49 BRKW2   LDA  PSTAT     ; ELSE CHOOSE CODE THAT WILL RESTORE
FF13:10 FB      50         BPL  BRKW2
FF15:A9 01      51         LDA  #$01      ; NON ZERO AND POSITIVE
FF17:D0 07      52         BNE  BRK1      ; BRANCH ALWAYS
FF19:           53 *
FF19:           54 *
FF19:AD F1 FF   55 FNEG    LDA  PSTAT     ; LOOK AT THE OK TO WRITE FLAG
FF1C:10 FB      56         BPL  FNEG       ; WAIT FOR IT TO GO HIGH
FF1E:A9 80      57         LDA  #$80      ; THIS WILL INC TO NEG
FF20:8D F0 FF   58 BRK1    STA  WDATA     ; SEND FLAG CODE
FF23:AD F1 FF   59 BRKW4   LDA  PSTAT     ; WAIT TO STORE X REG
FF26:10 FB      60         BPL  BRKW4
FF28:8E F0 FF   61         STX  WDATA     ; SEND THE X REGISTER
FF2B:AD F1 FF   62 BRK2    LDA  PSTAT     ; LOOK AT THE OK TO WRITE FLAG
FF2E:10 FB      63         BPL  BRK2      ; KEEP WAITING
FF30:8C F0 FF   64         STY  WDATA     ; SEND THE Y REGISTER
FF33:AD F1 FF   65 BRK3    LDA  PSTAT     ; CHECK FLAG AGAIN
FF36:10 FB      66         BPL  BRK3      ; AND WAIT FOR IT
FF38:BA         67         TSX             ; GET THE STACK POINTER
FF39:8E F0 FF   68         STX  WDATA     ; SEND IT OVER
FF3C:A2 E0      69         LDX  #RSAVE    ; SET UP LOOP TO SAVE RAM LOCATIONS
FF3E:AD F1 FF   70 BRK4    LDA  PSTAT     ; CHECK OK TO WRITE FLAG
FF41:10 FB      71         BPL  BRK4      ; LOOP ON IT
FF43:B5 00      72         LDA  0,X       ; GET RAM VALUE
FF45:8D F0 FF   73         STA  WDATA     ; SEND IT TO THE APPLE
FF48:E8         74         INX               ; MOVE TO NEXT
FF49:30 F3      75         BMI  BRK4
```

```
FF4B:           76 *
FF4B:CA         77          DEX             ; NOW RESET THE STACK POINTER
FF4C:9A         78          TXS
FF4D:08         79          PHP             ; PUSH THE FLAGS
FF4E:68         80          PLA             ; GET THEM IN A
FF4F:20 98 FF   81          JSR  WBA        ; GO SEND FLAGS
FF52:           82 *
FF52:           83 *
FF52:           84 * INITIALIZATION FROM RESTART
FF52:           85 *
FF52:           86 RESTART EQU  *
FF52:A2 FF      87          LDX  #$FF       ; SETUP STACK POINTER
FF54:9A         88          TXS
FF55:D8         89          CLD             ; CLEAR DECIMAL MODE
FF56:AD F2 FF   90          LDA  RDATA      ; CLEAR BUFFER
FF59:           91 *
FF59:           92 * FALL INTO COMMAND LOOP
FF59:           93 *
FF59:           94 *
FF59:           95 * THIS IS THE COMMAND LEVEL
FF59:           96 *
FF59:20 A1 FF   97 CI       JSR  RBA        ; GET A POSSIBLE COMMAND BYTE
FF5C:C9 10      98          CMP  #$10       ; DO WE READ MEM?
FF5E:F0 5B      99          BEQ  RM         ; IF SO DO IT
FF60:C9 20     100          CMP  #$20       ; DO WE WRITE MEM?
FF62:F0 66     101          BEQ  WM         ; IF SO DO IT
FF64:C9 30     102          CMP  #$30       ; DO WE GO FROM BREAK?
FF66:F0 07     103          BEQ  GO         ; IF SO DO IT
FF68:C9 40     104          CMP  #$40       ; DO WE JUMP INDIRECT?
FF6A:D0 ED     105          BNE  CI         ; IF NOT KEEP LOOPING
FF6C:4C D9 FF  106          JMP  GOI        ; ELSE DO THE JMP INDIRECT
FF6F:          107 *
FF6F:          108 * ROUTINE TO RESTORE STATE AFTER BREAK POINT AND JMP
FF6F:          109 * BACK INTO THE VCS PROGRAM
FF6F:          110 *
FF6F:          111 *
FF6F:AD F4 FF  112 GO       LDA  F1REST     ; GET PART 1 OF THE FLAG RESTORE PROCESS
FF72:48        113          PHA             ; PUT IN FLAGS BY WAY OF STACK
FF73:28        114          PLP
FF74:          115 *
FF74:A2 20     116          LDX  #256-RSAVE ; LOOP COUNT FOR RAM STUFF BACK
FF76:AD F1 FF  117 GO1      LDA  PSTAT      ; LOOK AT STATUS
FF79:29 40     118          AND  #$40       ; SEE IF STUFF THERE
FF7B:F0 F9     119          BEQ  GO1        ; IF NOT THEN WAIT
FF7D:AD F2 FF  120          LDA  RDATA      ; THEN GET VALUE
FF80:95 DF     121          STA  RSAVE-1,X  ; PUT BACK IN RAM
FF82:CA        122          DEX             ; MOVE TO NEXT
FF83:D0 F1     123          BNE  GO1        ; LOOP
FF85:          124 *
FF85:          125 *
FF85:AE F9 FF  126          LDX  SREST      ; GET VALUE IN X TO RESTORE STACK POINTER
FF88:9A        127          TXS             ; RESTORE STACK POINTER
FF89:AC F8 FF  128          LDY  YREST      ; GET VALUE TO RESTORE Y
FF8C:AE F7 FF  129          LDX  XREST      ; GET VALUE TO RESTORE X
FF8F:AD F6 FF  130          LDA  AREST      ; GET VALUE TO RESTORE A
FF92:EE F5 FF  131          INC  F2REST     ; RESTORE Z AND N FLAGS
FF95:          132 * NOW ALL IS RESTORED SO JMP BACK
FF95:4C 00 FF  133 BJUMP    JMP  BREAK
FF98:          134 *
FF98:          135 *
FF98:          136 *
FF98:          137 *
FF98:          138 * HERE IS THE COMM PACKAGE
FF98:          139 *
FF98:2C F1 FF  140 WBA      BIT  PSTAT      ; GET READY TO WRITE A BYTE TO THE APPLE
FF9B:10 FB     141          BPL  WBA        ; WAIT FOR WRITE OK FLAG
FF9D:8D F0 FF  142          STA  WDATA      ; THEN DO IT AND RETURN
FFA0:60        143          RTS
FFA1:          144 *
FFA1:          145 *
FFA1:2C F1 FF  146 RBA      BIT  PSTAT      ; GET READY TO READ SOMETHING
FFA4:50 FB     147          BVC  RBA        ; WAIT FOR READ READY BIT
FFA6:AD F2 FF  148          LDA  RDATA      ; READ THE DATA
FFA9:60        149          RTS               ; THEN DONE
FFAA:          150 *
FFAA:          151 *
FFAA:          152 *
```

```
FFAA:           153 * ROUTINE TO SET UP A 16 BIT ADDRESS AND 8 BIT COUNT
FFAA:           154 *
FFAA:20 A1 FF   155 GETADR  JSR  RBA       ; GO GET HIGH ADDRESS PART
FFAD:85 F9      156         STA  RBASE+1   ; PUT INTO BASE POINTER
FFAF:20 A1 FF   157         JSR  RBA       ; GET LOWER PART
FFB2:85 F8      158         STA  RBASE     ; PUT IN POINTER
FFB4:20 A1 FF   159         JSR  RBA       ; GET THE LENGTH
FFB7:AA         160         TAX             ; PUT IN X REG
FFB8:A0 00      161         LDY  #0        ; USE Y FOR INDEX
FFBA:60         162         RTS            ; NOW ADDRESS AND COUNT SETUP
FFBB:           163 *
FFBB:           164 *
FFBB:           165 *
FFBB:           166 * ROUTINE TO READ OUT MEMORY
FFBB:20 AA FF   167 RM      JSR  GETADR    ; SET UP ADDRESS AND COUNT
FFBE:B1 F8      168 RM1     LDA  (RBASE),Y ; GET VALUE
FFC0:20 98 FF   169         JSR  WBA
FFC3:C8         170         INY            ; MOVE INDEX
FFC4:CA         171         DEX            ; DECREMENT COUNT
FFC5:D0 F7      172         BNE  RM1       ; LOOP TILL DONE
FFC7:4C 59 FF   173         JMP  CI        ; DONE
FFCA:           174 *
FFCA:           175 *
FFCA:           176 *
FFCA:           177 * ROUTINE TO WRITE MEMORY
FFCA:           178 *
FFCA:20 AA FF   179 WM      JSR  GETADR    ; GO GET ADDRESS AND COUNT
FFCD:20 A1 FF   180 WM1     JSR  RBA       ; GO GET BYTE
FFD0:91 F8      181         STA  (RBASE),Y ; STORE IT
FFD2:C8         182         INY            ; MOVE INDEX
FFD3:CA         183         DEX            ; DECREMENT COUNT
FFD4:D0 F7      184         BNE  WM1       ; LOOP TILL DONE
FFD6:4C 59 FF   185         JMP  CI        ; DONE
FFD9:           186 *
FFD9:           187 *
FFD9:           188 *
FFD9:           189 * ROUTINE TO TAKE A JMP INDIRECT
FFD9:           190 *  (USED TO PUT THINGS TO SLEEP)
FFD9:20 A1 FF   191 GOI     JSR  RBA       ; GET HIGH ORDER JMP ADDR
FFDC:85 F9      192         STA  RBASE+1   ; PUT IN POINTER HIGH
FFDE:20 A1 FF   193         JSR  RBA       ; GET LOW ORDER JMP ADDR
FFE1:85 F8      194         STA  RBASE     ; PUT IN POINTER LOW
FFE3:6C F8 00   195         JMP  (RBASE)   ; GO THERE
FFE6:           196 *
FFE6:           197 * PATCH AREA
FFE6:00 00      198         DW   0
FFE8:00 00      199         DW   0
FFEA:00 00      200         DW   0
FFEC:00 00      201         DW   0
FFEE:00 00      202         DW   0
FFF0:           203 *
FFF0:           204 *
FFF0:           205 *
FFF0:           206 * TOP OF MEMORY STUFF -- THIS ADDRESS MUST BE $FF00
FFF0:           207 *
FFF0:           208 * THESE REGISERS ARE USED FOR
FFF0:           209 * COMMUNICATION TO AMON -- SO DONT REUSE THIS AREA
FFF0:           210 *
FFF0:00 00      211         DW   0
FFF2:00 00      212         DW   0
FFF4:00 00      213         DW   0
FFF6:00 00      214         DW   0
FFF8:00 00      215         DW   0
FFFA:59 FF      216         DW   CI        ; REENTRY VECTOR
FFFC:52 FF      217         DW   RESTART   ; RESET VECTOR
FFFE:00 00      218         DW   0         ; END OF ADDRESS SPACE (FFFE)
0000:           219 *
0000:           220 *
0000:           221 *
0000:           222 * END OF FMON

\007*** SUCCESSFUL ASSEMBLY: NO ERRORS
```

```
 FFF6 AREST         ?FF95 BJUMP          FF00 BREAK          FF20 BRK1
 FF2B BRK2           FF33 BRK3           FF3E BRK4           FF05 BRKW1
 FF10 BRKW2          FF23 BRKW4          FF59 CI             FFF4 F1REST
 FFF5 F2REST         FF19 FNEG           FFAA GETADR         FF6F GO
 FF76 GO1            FFD9 GOI            FF0E NOTZ           FFF1 PSTAT
 FFA1 RBA              F8 RBASE          FFF2 RDATA          FF52 RESTART
 FFBE RM1            FFBB RM               E0 RSAVE          FFF9 SREST
 FF98 WBA            FFF0 WDATA          FFCA WM             FFCD WM1
 FFF7 XREST          FFF8 YREST
```

```
   E0 RSAVE           F8 RBASE          FF00 BREAK          FF05 BRKW1
 FF0E NOTZ          FF10 BRKW2          FF19 FNEG          FF20 BRK1
 FF23 BRKW4         FF2B BRK2           FF33 BRK3          FF3E BRK4
 FF52 RESTART       FF59 CI             FF6F GO            FF76 GO1
?FF95 BJUMP         FF98 WBA            FFA1 RBA           FFAA GETADR
 FFBB RM            FFBE RM1            FFCA WM            FFCD WM1
 FFD9 GOI           FFF0 WDATA          FFF1 PSTAT         FFF2 RDATA
 FFF4 F1REST        FFF5 F2REST         FFF6 AREST         FFF7 XREST
 FFF8 YREST         FFF9 SREST
```